Embedded Systems Design
Kimonidis Alexandros
1st Lab

1.2.1

OS: cat *etc*/os-release

fedora 32

Kernel: uname -r

5.9.8-100.fc32.x86_64

Cache Memory Hierarchy: getconf -a | grep CACHE

LEVEL1_ICACHE_SIZE	32768
LEVEL1_ICACHE_ASSOC	8
LEVEL1_ICACHE_LINESIZE	64
LEVEL1_DCACHE_SIZE	32768
LEVEL1_DCACHE_ASSOC	8
LEVEL1_DCACHE_LINESIZE	64
LEVEL2_CACHE_SIZE	262144
LEVEL2_CACHE_ASSOC	4
LEVEL2_CACHE_LINESIZE	64
LEVEL3_CACHE_SIZE	12582912
LEVEL3_CACHE_ASSOC	16
LEVEL3_CACHE_LINESIZE	64
LEVEL4_CACHE_SIZE	0
LEVEL4_CACHE_ASSOC	0
LEVEL4_CACHE_LINESIZE	0

RAM Memory: sudo dmidecode --type 17

Handle 0x0039, DMI type 17, 40 bytes
Memory Device
Array Handle: 0x0038
Error Information Handle: Not Provided
Total Width: 64 bits
Data Width: 64 bits
Size: 8 GB
Form Factor: SODIMM
Set: None
Locator: DIMM A
Bank Locator: Not Specified
Type: DDR4
Type Detail: Synchronous
Speed: 2667 MT/s
Manufacturer: 80AD000080AD
Serial Number: 2F74FFFC
Asset Tag: 02192800

Part Number: HMA81GS6JJR8N-VK
Rank: 1
Configured Memory Speed: 2667 MT/s
Minimum Voltage: 1.2 V
Maximum Voltage: 1.2 V
Configured Voltage: 1.2 V

Handle 0x003A, DMI type 17, 40 bytes
Memory Device
Array Handle: 0x0038
Error Information Handle: Not Provided
Total Width: 64 bits
Data Width: 64 bits
Size: 8 GB
Form Factor: SODIMM
Set: None
Locator: DIMM B
Bank Locator: Not Specified
Type: DDR4
Type Detail: Synchronous
Speed: 2667 MT/s
Manufacturer: 80AD000080AD
Serial Number: 2F750008
Asset Tag: 02192800
Part Number: HMA81GS6JJR8N-VK
Rank: 1
Configured Memory Speed: 2667 MT/s
Minimum Voltage: 1.2 V
Maximum Voltage: 1.2 V
Configured Voltage: 1.2 V

Core Count/Frequency: cat /proc/cpuinfo

Output was too big.

In summary:

Core count: 16

Core Frequency: 2.6 Ghz

1.2.2

Used gettimeofday function before and after running the function to get the execution time in the **phods_original.c** file.

1.2.3

The changes in the code include:

- Joining the loop that initializes the vector_x and vector_y arrays with the loop that does the calculations
- Keeping in temp variables the calculations accesses the reoccur:
  - temp[0]=B*x
  - temp[1]=tempo[0]+k
  - temp[2]=temp[1]+vectors_x[x][y]+i
  - temp[3]=temp[2]-i

1.2.4

Looped B through all the common divisors of M and N.

Changed B's value in the source code by using the sed command on #define B.

Ran each configuration 10 times to get the average and kept the best average.

1.2.5

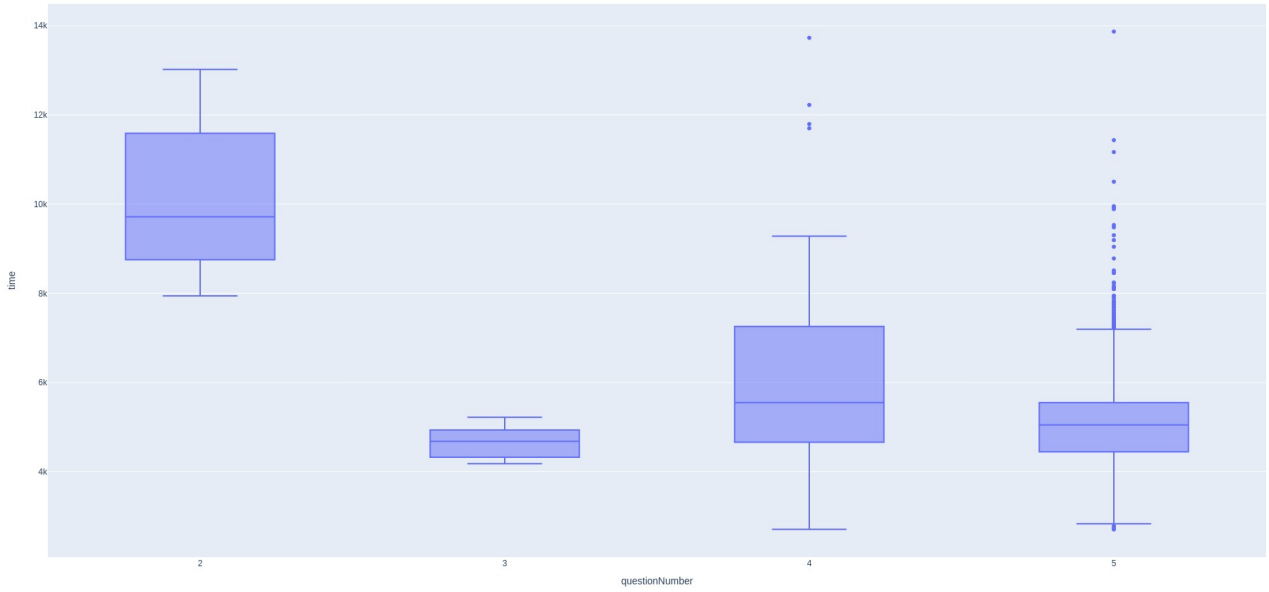
Substituted #define B with #define Bx and By.

Also substituted B in the source code with the corresponding for the loop Bx or By.

In the script I looped Bx and By through the divisors of N and M accordingly and ran each configuration for 10 times to get the average.

In the end I kept the configuration that ran the fastest.

1.2.6



As we can see in the box plots:

- We have the worst time in the unoptimized source code.
- Better time in the optimized code of question 3.
- In the box plot of question 4 we can see that we have even better run times with the right configuration but we can have really bad run times with bad configurations.
- At question 5 we can see that we have pretty much the same run time as question 4. There's not much use breaking the block into unequal x and y sizes

1.3.1

I added the gettimeofday function before and after the execution of the critical code so as to be able to get the execution time.

Afterwards I ran the program 10 times and got the average execution time which equaled **502153 usec**.

1.3.2

Exhaustive Search

Unroll Factor: 7

Execution Time: 285476 usec

Simplex

Unroll Factor: 21

Execution Time: 304091 usec

Random Search

Unroll Factor: 16

Execution Time: 300380 usec

1.3.3

As we can from the above results the time of the unoptimized program is around 500k usecs which is the worst of all the other times as expected.

Afterward we have exhaustive that gives us the best time, which is expected since it tried all the combinations to get the best one.

And last we have simplex and random which in my run had almost the same times. Of course the problem with random search is it may not always give such good results (ex. I had a random search that gave me runs in the 40k range).